Proceedings of the Federated Conference on Computer Science and Information Systems pp. 511–519
DOI: 10.15439/2020F48
ISSN 2300-5963 ACSIS, Vol. 21

A Federated Platform to Support IoT Discovery in Smart Cities and HADR Scenarios

Lorenzo Campioni[1], Niccolò Fontana[1,2], Alessandro Morelli[2], Niranjan Suri[3,2], Mauro Tortonesi[1]
[1] Distributed Systems Research Group, University of Ferrara, Ferrara, Italy
{lorenzo.campioni, mauro.tortonesi}@unife.it
[2] Florida Institute for Human and Machine Cognition (IHMC), Pensacola, FL, USA
{amorelli, nfontana, nsuri}@ihmc.us
[3] US Army Research Laboratory (ARL), Adelphi, MD, USA
niranjan.suri.civ@mail.mil

***Abstract*—Smart Cities are among the most dynamic and rapidly evolving modern environments, driven by the development of new technologies and the fast growth of the Internet of Things (IoT), which enable the acquisition and processing of very large amounts of data. However, accessing IoT assets is proving to be a challenge, as neither formal nor de facto standards to discover connected Things have emerged. Services that provide discovery and access capability for IoT resources are in the rise, but they often adopt service-specific interfaces and authorization mechanisms that hinder the development and maintainability of IoT applications. Low flexibility and interoperability become especially problematic during emergency situations, when responders might need to access resources that normally would not be allowed to access. To address these issues, this paper describes MARGOT, a distributed edge computing platform that supports domain-aware and secure discovery of IoT resources in Smart Cities. Experimental results obtained using MARGOT in an emulated network environment show that our platform can effectively reduce discovery latency and bandwidth consumption under the considered use cases and network conditions.**

***Index Terms*—Internet of Things (IoT), Humanitarian Assistance and Disaster Recovery (HADR), Resource Discovery, Distributed Information Systems**

I. Introduction

SMART Cities worldwide are thriving and evolving at great speeds, mainly driven forward by the possibility of combining innovative Information and Computing Technology (ICT) solutions with small, cheap, and yet powerful computational and sensor technology, which has paved the way for new business opportunities that are attracting large numbers of investors and industry leaders to the sector. Leveraging ICT to process large amounts of data generated by connected sensors, actuators, and other intelligent objects that are part of the Internet of Things (IoT), Smart Cities aim at improving their citizens' quality of live by enhancing government, health, transportation, security, education, and other services [1]–[4].

The accessibility of "Things" and their interoperability with other systems are major challenges that Smart Cities have to face, as they directly impact their capability to advance at a fast pace by introducing new services or extending the ones currently offered [5]. To address this issue, public and private organizations that work in the field of smart services and IoT have been developing new services that allow applications to retrieve access information on smart Things and other connected devices managed by those organizations. In the future, with the development of new communications and computation technologies, which enable the autonomous discovery of new nodes and resources in the network, and the definition of formal standards for the IoT, we expect that more and more players will enter the market and the number of IoT services and connected devices to rise as a consequence.

These IoT services are typically Cloud-based [6], [7], offer a proprietary Application Programming Interface (API) that reflects the nature of exposed assets, and employ a multitude of different techniques to enforce security and verify clients' authorization [8]. This strongly reduces accessibility and interoperability and raises the need for solutions that can simplify the discovery and access of the IoT assets exposed by those heterogeneous services. Proposed approaches will need to match domain-specific security requirements, offer good performance to users and applications, and avoid taking a heavy toll on the Smart City's network infrastructure. In addition, those approaches will have to be able to support the cities' administration in case of Humanitarian Assistance and Disaster Relief (HADR) missions and operations, during which the network connectivity might be limited and it is vital that emergency responders are able to access critical data and resources from the stricken areas.

As a step in this direction, the present paper describes the design and architecture of MARGOT, a platform to support domain-aware and context-aware discovery of IoT resources in Smart Cities. This work extends our previous study in [9] by discussing the use of Federation Services [10] to provide an array of distributed capabilities within MARGOT. We also present the results of new experiments that demonstrate MARGOT's ability to lower IoT asset discovery latency for applications and reduce network bandwidth consumption under specific usage and network conditions.

II. Accessing IoT resources in Smart Cities

Smart Cities and other smart environments are characterized by the extensive and effective use of ICT solutions to improve human day-by-day activities, enabling smart living and the development and deployment of next generation services.

In particular, Smart Cities take advantage of the pervasive presence of connected ”intelligent objects” that interact with the environment. This capillary network of IoT devices permits to acquire large quantities of data on their surroundings, e.g. sensor measurements, images, audio, video, and so on, that can be processed and then made available to users and applications. Such capabilities enable the development of time-critical, context-, and location-aware services for the smart citizens.

However, despite the IoT being one of the main pillars of Smart Cities, it also poses several challenges, which include the following: IoT devices present computational and power limitations that might reduce accessibility; small sensors and other Things typically only support specific communication protocols for constrained devices; multiple actors are involved, as owners and administrators of the devices, with different security and access policy requirements. Given the incredibly large and continuously growing number of connected Things and their extreme heterogeneity, in terms of data produced, location, domain, accessibility, and others, it becomes crucial to provide solutions that enable users and applications to identify resources based on specific requirements and interests.

To this day, several cities and organizations have deployed solutions to enable users and applications to obtain access information about IoT resources made available within the domain (e.g., the city). Such solutions are typically Cloud-based and offer a web-based API, generally encoded using JSON or other standard formats, that allows consumers to query for access and other types of information about the managed IoT resources.

These IoT services typically allow to search for different types of devices, including webcams, weather sensors and stations, several kinds of sensors (pollution, noise, light, traffic, etc.), vehicles and other transportation-related things, and so on. Some of these services are: Windy (https://www.windy.com), OpenWeather (https://openweathermap.org), City Bikes (https://citybik.es), Thingful (https://www.thingful.net), Digitraffic (https://www.digitraffic.fi), the New York State’s 511 Traveler Information System (511NY) (https://511ny.org), LookCAM (https://lookcam.com), and Airly (https://airly.eu). Some services do not offer direct access to the managed devices, but provide an interface to query and retrieve the data generated by them.

For this study, we used data obtained from the services provided by Digitraffic, NY511, and Airly. The Finnish Transport Agency operates Digitraffic, which offers real time traffic information that covers road, marine, and rail traffic in Finland. NY511 (https://511ny.org), hosted by the State of New York, provides information tightly related to transportation services and road conditions throughout the State. Finally, Airly gathers information related to air quality around the globe using sensors that measure the concentration of PM1, PM2.5, PM10, and NO2 and O3 gases in real time.

III. Future IoT Services and HADR Scenarios

Since the term IoT was coined, the population of devices that pervade smart environments has constantly grown. Thanks to The multitude of novel communication solutions, such as LTE/4G, 5G, LoRa, LoRaWAN, and other lightweight communication protocols, more and more devices have been able to connect to the Internet and generate information that other devices and applications can consume. Scientists expect that the number of devices connected to the Internet will continue to grow strongly in the next years [11], sustained by the decreasing cost of hardware, the development of new technologies more suited to constrained devices, and the definition of official standards, such as the IEEE P2413 ”Standard for an Architectural Framework for the Internet of Things (IoT)” [12], that can simplify the interaction between devices and other network actors. Furthermore, numerous private organizations and stakeholders have been attracted by the business opportunities involved with Smart Cities and IoT and started to deploy their own private sensors, thus actively participating to the growth of the number of IoT devices connected.

This growth further increases the need for services akin to those provided by Digitraffic, 511 NY, and Airly. As an example, let us consider an application for police authorities that makes use of traffic camera feeds and image recognition software in order to track down a criminal during a chase. The application requires access (IP addresses, protocols used, and so forth) and other information (e.g., cameras’ locations) to be able to connect to cameras located in positions from which they can record the target’s movements. Without other solutions, the application must have prior knowledge of all cameras’ locations and access information, an approach that is not really suited for highly dynamic environments such as Smart Cities, where sensors can move, sleep, or fail for a number of possible reasons. In fact, this approach would require that either all these events are notified to all applications or applications are designed to handle failures nicely and fallback to other data sources (for instance, if the desired camera is offline, there might still be a lower resolution camera that can record the same area, or other cameras nearby that might still provide useful streams for the purpose of the application). Furthermore, the Smart City infrastructure is typically composed of many different domains where resources are managed and maintained by different organizations, each with potentially different security and access policies. IoT discovery services can help mitigate the complexity of discovering and then accessing IoT devices in multi-domain Smart City scenarios by providing updated information on live resources to applications, supporting sophisticated search criteria to ensure that clients are informed about all relevant resources, and requiring the authentication of clients.

We envision that the Smart Cities’ highly dynamic environment and the constant growth of connected IoT assets will lead to a new generation of IoT services that are able to autonomously discover and register new resources in the

network. Such services will significantly simplify and speed up the deployment of new devices, since they will not require manual registration to be discoverable by clients. In addition, these new IoT services will provide all stakeholders with a constantly updated view of the status of the devices within the environment and information on how to retrieve data from them, e.g., via a M2M-compliant API.

These services will also offer a strategic advantage during HADR operations in Smart City environments. HADR operations take place after a disaster has severely damaged parts of the environment and/or put human lives at risk, which requires local authorities to immediately enact safety protocols to assists the victims and prevent aggravating the current situation. During these procedures, the ability to exploit local sensors and other IoT assets plays a key role in increasing the effectiveness of HADR operations by improving the situational awareness of responder teams. Moreover, the support for the automatic discovery of new resources allows those teams to deploy sensors on-the-fly (e.g. a camera-equipped drone) whenever required and retrieve the produced data using the same applications they would use to access other devices in the city.

However, the Smart City network infrastructure could also suffer damages during disasters. As a consequence, links can be severed and nodes can break, causing traffic to be re-routed, network congestion on the unstruck links to worsen, and portions of the network to become unreachable. The IoT infrastructure is especially impacted when it requires connectivity to the Cloud, as it often happens with sensors that publish collected data to remote servers or when applications periodically check the status of IoT devices by means of some form of network polling. In these conditions, IoT services and applications that run within the edge network and are able to autonomously discover available resources would offer a more robust solution. More specifically, the automatic discovery of IoT resources would enable services and applications to identify assets that have become unavailable, clients might still be able to connect to the desired nodes located in the same edge network, and emergency responders could still deploy and access new sensors dynamically during a mission.

Despite the advantages that we expect new generation IoT services to bring about, many challenges will remain. First, future IoT services will likely share information through proprietary APIs that satisfy domain-specific requirements. As a consequence, IoT applications that allow emergency responders and other personnel to access assets across multiple domains will be required to implement and maintain different interfaces to each service. From the security perspective, these software will often have to support and manage several security protocols, authentication mechanisms, and user certificates to be compatible with the requirements of different services, which require considerable coding efforts and costs. During HADR and other emergency operations, rescue teams might need to access devices that would not normally be allowed to; to address this requirement, special solutions will likely be needed that necessitate additional work from both software developers and IoT administrators responsible for the different domains. Moreover, developers of IoT applications for emergency responders will have to invest significant efforts to design software that can withstand partial network failures, for instance implementing distributed caching, supporting peer-to-peer querying and data retrieval, and discovering new sensors on-the-fly.

IV. The MARGOT Platform

Due to the considerations discussed in Section III, solutions able to locate new IoT resources autonomously and designed to be more robust in presence of HADR situations are extremely interesting. In this context, we developed a distributed edge computing platform, MARGOT, that has the goal of simplifying the development of IoT applications. MARGOT permits to discover IoT resources across separated domains and network segments and supports context-aware applications by providing them with a query interface that accepts parameters to refine the search criteria. IoT applications can interact with MARGOT via a JSON RESTful API. The architecture of MARGOT is represented in Fig. 1, which shows the major components, i.e., the Discovery Agents (DAs), the Information Processor (IP), Federation Services (or Information Management System Bridge, IMSBridge), and the ReST API, and the interactions between them. The MARGOT platform is designed in such a way that each instance is responsible for the discovery of resources within one or more domains, and each domain has one and only one MARGOT instance of reference, which will typically be deployed within the same network or close to it, i.e., geographically and/or in terms of network hops. The knowledge of all discovered IoT resources is distributed across all MARGOT instances in the system, which exchange information via Federation Services (we sometimes refer to the set of MARGOT instances connected via Federation as ”federated MARGOT instances” or ”MARGOT federation”).

DAs implement domain-specific resource discovery and store data about the discovered assets in the local MARGOT database. As shown in Fig.1, DA implementations rely on specific protocols, such as MQTT or CoAP, and the corresponding discovery procedures to detect and identify IoT assets within the local network or domain. To support the discovery of the resources made available via services like 511NY or Airly, it is enough to write a DA that implements the API of the chosen service. DAs are also responsible for complying with any security requirement imposed by the domains. For instance, an IoT domain could require DAs to be authenticated and authorized before they can access the domain resources.

DAs can perform the discovery process either proactively, if the process is executed periodically or the implemented protocol supports some form of proactive discovery, or reactively, when specific events trigger discovery, such as a request issued by a client or coming from a federated MARGOT instance. Generally speaking, proactive behavior trades query latency for the freshness of the information. Depending on the rate of user requests and the cost of the discovery process, proactive discovery could lead to either a decrease or an

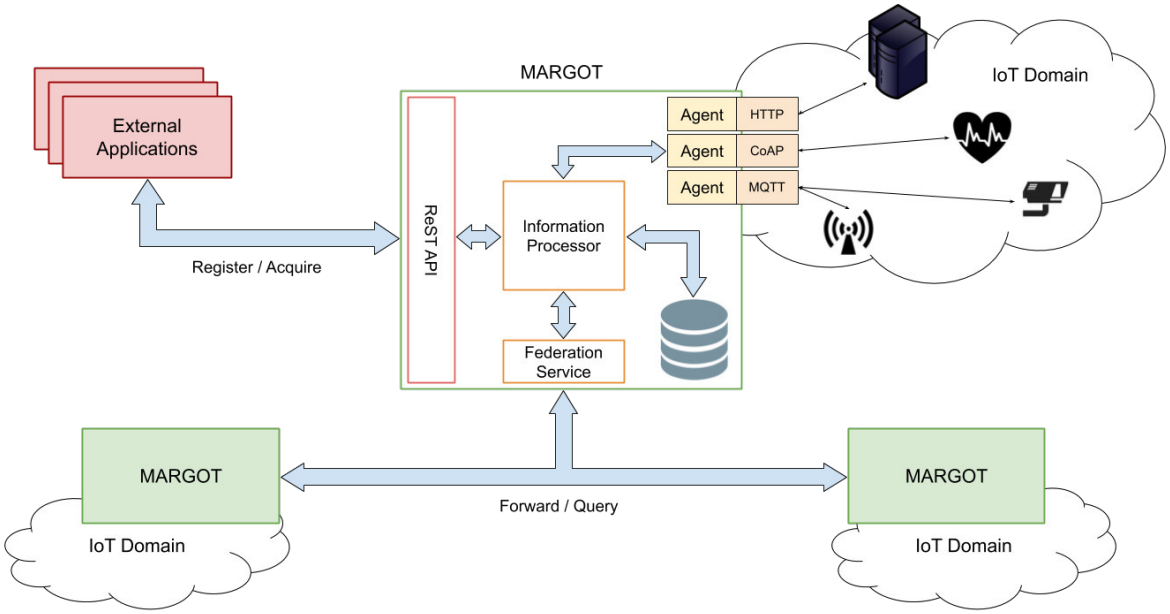

Fig. 1. The MARGOT Architecture

increase in bandwidth consumption. To tackle this matter and provide more efficient proactive discovery strategies, DAs should allow MARGOT to tune parameters that control the discovery process, e.g., the frequency of the process. Protocols that naturally support proactive discovery, such as CoAP and MQTT, can increase the efficiency of detecting new assets, but they still require some form of periodic probing to ensure that previously discovered Things are still alive and reachable.

The IP takes care of processing the data received from the DAs, storing them in the local database, and handling clients' requests. While doing this, the IP also collects statistics that characterize the domain and user requests, including the variability of the discovered IoT assets and frequently requested resources, and merges them with the statistics received from federated MARGOT instances. Finally, the IP manages data exchange via Federation Services: it decides which IoT resources to publish into the MARGOT federation, forwards user queries if necessary, replies to queries received from remote MARGOTs, and shares updated domain statistics.

The IP is also responsible for tuning the behavior of proactive DAs and perform other optimizations based on the acquired statistics. For instance, in presence of highly variable local domains, MARGOT will typically request registered DAs to increase the frequency of discovery, adjust its caching policy by decreasing the expiration time for the resources in those domains, and notify federated MARGOTs about the changes. This will affect the number of user queries that will be forwarded to federated MARGOTs against the number of requests that will be resolved using the data cached in the local database, which in turn will have an impact on the system bandwidth utilization and the accuracy of the information returned to clients.

MARGOT also supports *proactive querying*, which guarantees that the information cached in its local database about certain IoT assets is always up-to-date. MARGOT activates this mechanism if the number of user requests for the same set of resources in a given period of time is above a *"trigger"* threshold and preserves it until that number falls below a *"maintenance"* threshold. Both thresholds are configurable and can be tuned by the local MARGOT administrator. When proactive querying for a certain set of resources has been activated, whenever one of the DAs reports an update to at least one element of the set, MARGOT automatically pushes the updated information to all federates. By doing this, other MARGOT instances will be able reply to user requests that involve any elements in the set directly, without having to send queries to other federates. Proactive querying affects the amount of data exchanged by MARGOT via Federation in a way that ultimately depends on the number of assets in the set, how often they are updated, the number of MARGOT instances involved, and the amount of user requests received that can be resolved from the resources in the set.

A. Data Sharing via Federation Services

MARGOT relies on Federation Services [10] to exchange data with other MARGOT instances. Federation Services provide clients of the IMSBridge (federates) with a completely distributed publish-subscribe communications infrastructure that supports and simplify information exchange in multi-domain scenarios. *Topics* control the routing of data over the Federation network. Topics are named abstractions (i.e., identified by unique strings) that can be thought of as independent communication channels over the network; some topics are typically predefined via configuration files, but federates can also create new ones dynamically at run-time. New messages published within one topic are delivered to all federates sub-

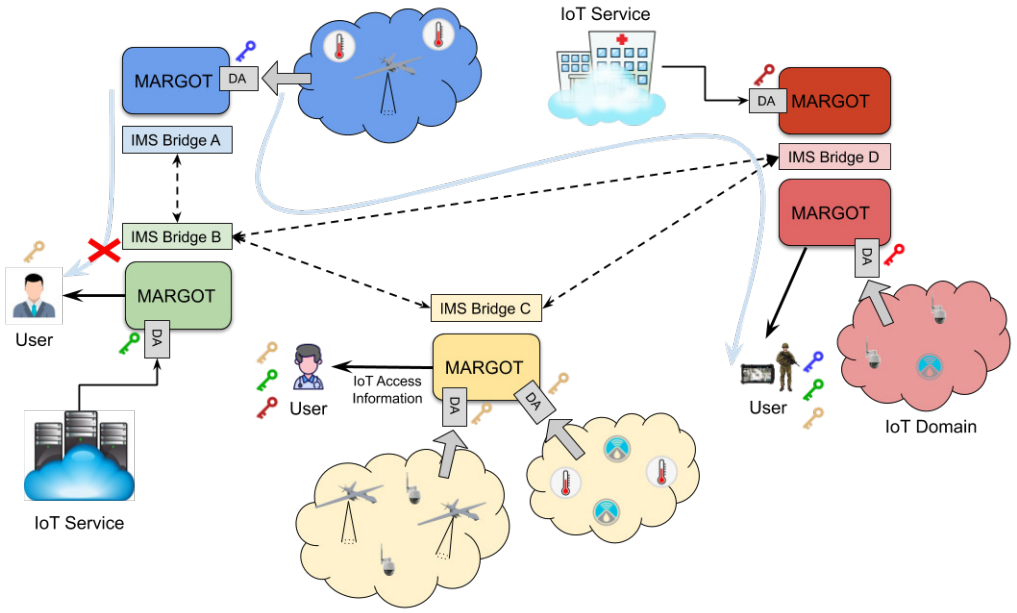

Fig. 2. MARGOT with Federation Services and ABE security

scribed to that topic, which will receive the message directly from the publisher or from another IMSBridge. Clients can join and leave these channels at any time.

Each IMSBridge instance can specify policies that determine what information it is allowed to share with every other instance. This allows one domain's administrators to define rules locally that implement that domain's security and data sharing policies, without the need to coordinate with other domains' administrators. Sharing policies can be enforced using a security system based on Attribute-based Encryption (ABE), which permits to combine different security layers on a per-message basis [13] and ensures that only clients in possess of the right keys can access the messages. Federates can use ABE to encrypt the payload and metadata of messages before publication, but the information relative to the topic of publication will not be encrypted to guarantee correct routing.

Federation Services support other useful features for distributed multi-domain environments. *Distributed queries* allow federates to send and run queries on all or a subset of IMSBridge instances, in order to retrieve published messages that match client-defined criteria. Distributed queries take advantage of metadata information specified during publication to select relevant messages. *Smart synchronization* enables two IMSBridge instances to exchange updated messages after a disconnection period, e.g., caused by network-problems or other issues. Depending on the nature of the messages, all or only the most recent updates will be exchanged.

MARGOT leverages Federation Services' capabilities for all distributed actions, including the discovery of new instances, proactive querying and IoT data replication on federated instances, remote query execution, and the exchange of control information and usage and domain statistics. This enables users and clients to discover resources available across multiple IoT domains and allows the platform to adapt its behavior, e.g., concerning caching and query forwarding, under certain circumstances. Finally, MARGOT leverages Federation Services' policy-based data sharing and ABE security to control the access to IoT assets information.

Figure 2 depicts a distributed MARGOT deployment connected via Federation Services. Each MARGOT in the Figure is connected to a single IMSBridge that enables it to exchange data with other MARGOT instances through the Federation network; dashed arrows represent connections between IMSBridge instances that compose the Federation network. Users can connect to any MARGOT instance to obtain access information about IoT assets discovered by any federated MARGOT, in accordance with the data sharing and security policies implemented by the IMSBridge instances involved. For example, on the right side of the Figure, a user with permissions to access IoT asset information for three different domains (represented by the blue, green, and yellow keys next to him) connects to its local MARGOT instance (light red box in the Figure) to retrieve information about sensors in the blue domain. MARGOT translates the request received from the user into a Federation query, which is disseminated from *IMS Bridge D* across the whole Federation network, solved locally by each federate, and finally the answers are relayed back to the node from which the query originated. MARGOT then caches the information locally for future requests and generates a response for the user with the requested information. As a second example, another user that only has permissions to access information about Things located in the yellow domain also issues a request for assets in the blue domain; however, this user does not have permissions to access IoT information

from the blue domain and so he or she will not be able to decrypt the data received from MARGOT.

B. What MARGOT brings to the Table

The MARGOT platform offers a number of extremely interesting features to IoT application developers, which significantly facilitate and speed up development. Without doubt, one of the most appealing functionalities is that MARGOT grants access to all discovered IoT resources via a single API. This feature can help cutting down development and maintenance costs of applications tremendously. Additionally, the API offered by MARGOT already provides the possibility to formulate selective queries to filter out unwanted resources, for instance restricted to a specific geographical area, domain, and/or sensor type. As a consequence, developers do not have to write the code to handle filtering (or, at least, that code can be simplified considerably) and applications will save bandwidth and battery life by downloading and processing less data, which is especially good for mobile applications. Finally, since MARGOT instances will generally be deployed in locations that enable the discovery of new IoT assets (think about the case of sensors whose discovery necessitates the use of multicast over the local network segment), MARGOT will be able to give users and applications access to resources that they would not be able to discover otherwise, because of network and protocol limitations.

MARGOT can also help IoT service providers at multiple levels. First, MARGOT's caching capability can reduce the amount of traffic that service providers will need to handle. Moreover, the local MARGOT instance will typically be deployed closer, e.g., in terms of network hops, to IoT resources and service providers than users, whose location cannot be predicted or controlled easily, and lower distances tend to increase network efficiency. Finally, MARGOT essentially decouples users' requests for information on IoT resources from their discovery; this allows discovery-related traffic to become independent from the number of system users, thus generating more stable and predictable traffic loads over the sensor networks. All this translates into lower costs for infrastructure, network service, and power consumption for providers.

MARGOT can also help the Smart City administrators by simplifying the management of permissions required to access resource discovery for different domains and IoT services. We can imagine that many services that will offer access to IoT assets in future Smart Cities will require users to authenticate before being able to call their API. This is already the case today, with services like Airly, which require users to acquire an API key to pass to each call to enforce service throttling and ensure that the requesting users have the right permissions. With the rise in the number of IoT services and domains that will require authentication, managing permissions will grow increasingly complex for both users and service providers. Thanks to the combined use of MARGOT and Federation Services, it becomes possible to offload some of the complexity to the platform. For instance, they will be able to create separated federations of trusted MARGOT instances within which information can be shared securely without any external access. This would allow city officials, law Enforcement, or emergency response personnel to share access information to IoT infrastructure segments managed by the different administrations without limitations through a dedicated and secured MARGOT federation.

Finally, MARGOT can help during certain HADR situations by mitigating some of the effects of partially unavailable network infrastructures, e.g., due to damage or power failure. In these scenarios, it might become impossible for users to reach the servers of a provider, such as 511NY, but they may still have access to part of the edge and sensor networks. A distributed solution like MARGOT, which replicates data across federates, enhances the whole system's fault tolerance by leveraging redundant instances running in dispersed geographical locations. Therefore, clients are more likely to still have their queries resolved even when the network infrastructure is partially down because they can issue requests to remote MARGOT instances that were unaffected by the disaster. Once an instance receives a client request, even if the MARGOT with the responsible DA remains unreachable, it can still respond with the requested data via Federation, as long as at least one federate has cached those data in the past.

V. Experimental Results

We performed three different experiments to evaluate the effectiveness of a Federation of MARGOT in a multi-domain scenario; for an evaluation in a single domain context, the reader can refer to [9]. We conducted the experiments in an emulated network environment created using the Extensible Ad-hoc Networking Emulator (EMANE), which allows to control latency, bandwidth, and packet loss of the emulated network links. The scenario consists of 3 separated networks, which we will call Domain *A*, *B*, and *C*, respectively, connected via EMANE-controlled links. More specifically, the link between Domain *A* and Domain *B* presents a latency of 30 ms; Domain *A* is connected to Domain *C* via a 80 ms latency link; finally, the link that connects Domain *B* and Domain *C* has a latency of 100 ms.

Each domain has a node running MARGOT that can acquire information about IoT resources that the instances running in the other domains cannot directly obtain. To do so, we configured a DA in each MARGOT to interface with one of the three IoT services that we described in section II: NY511, Airly, and Digitraffic. In addition, each MARGOT is connected to a local IMSBridge that can federate with the other bridges via the EMANE-controlled links. As the results will show, MARGOT clients are able to connect to any MARGOT instance and retrieve information about IoT resources from any domain, as all MARGOT instances are federated.

In our scenario, three clients, which represent three rescue teams (Rescue Team 1 through Rescue Team 3), move from one domain to another and periodically query the local MARGOT instance to retrieve access information about devices that satisfy their interests. For simplicity reasons, each client

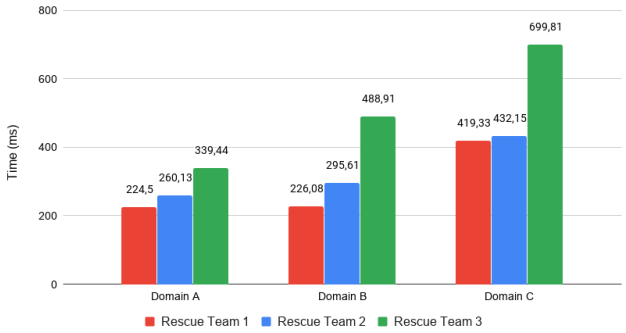

Fig. 3. Average Response Latency per Domain, MARGOT Caching Disabled

issue requests for the same subset of sensors each time: the first team generates requests for 10 sensors, the second team receives information about 50 sensors, and the third one about 250 sensors. We ran the first two experiments 100 times for each combination of Rescue Team and Domain, measuring the average response time from the client application. For the third experiment, we ran two tests of the duration of about one hour each, the first time with proactive query disabled and the second time after enabling that feature.

In the first experiment, we measured the average time that each rescue team in each domain had to wait to receive all the requested data without any use of caching within MARGOT; the goal was to measure the expected latency whenever new resources are queried for the first time or the cached entries are expired. As a consequence, for this experiment all MARGOT instances always forward the clients' queries to the other domains via Federation. Figure 3 shows the measured latency for each team receiving data from each domain. The first rescue team register the lowest latency since their requests involve less traffic, but the growth is less than linear (the number of assets queried increases five-fold when going from Rescue Team 1 to Rescue Team 2, and from Rescue Team 2 to Rescue Team 3, but the measured latency only increases by a fraction of that, with 65.4% being the highest increase), which suggests that the processing and forwarding of the queries has the largest impact on the total latency. Moreover, all teams present a higher average latency when connected to Domain *C*; this happens because the local MARGOT database contains a much larger number of entries compared to the other MARGOT instances, which increases the processing time.

The second experiment is similar to the first one, but in this case we primed the database of all MARGOT instances with the necessary data and then configured them to solve all clients' queries from the local database. The results, shown in Fig 4, are particularly relevant for scenarios in which clients request information about "popular" resources, which would often be resolved from cache, and in cases where the requested resources are fairly static and, consequently, high cache validity timeouts have been set for those resources. As the Figure shows, caching allows the system to reduce response times by one order of magnitude. As the response latency decreases, now ranging between 10 and 20 milliseconds, other factors, such as delays in the communications caused by the TCP protocol, thread scheduling and context switches, memory access, and others, whose impact in the first experiment was negligible, now have a visible effect on the measured response times. After examining the collected data, we concluded that they are the cause of the increased delay experienced by Rescue Team 1 and 3 when connected to Domain A and B, respectively.

We designed the final experiment to show the effects of MARGOT's proactive querying on bandwidth consumption. This experiment only involves Domain *A* and Domain C: we instantiated two static clients in Domain *A* that send a total of 5 requests per minute to retrieve data on the same subset of IoT resources located across the two domains, for a total of about 150 sensors. We repeated the experiment twice: in the first run, proactive querying was disabled (reactive querying), whereas, in the second run, we changed the configuration of MARGOT in Domain A to enable it (proactive querying). In the second run, after about 10 minutes, the frequent requests for the same set of resources trigger proactive querying, which enables the MARGOT running in Domain A to send a special request to the MARGOT in Domain C, after which the receiving MARGOT will start to push proactively any update regarding any of the resources that have been requested frequently.

The results obtained are shown in Figure 5. The graph shows the amount of traffic generated by the two MARGOT instances every minute with (in red) and without (in blue) the proactive querying optimization enabled. After the first 10 minutes and until one of the clients starts making requests for different sensors (which happens about 40 minutes after the beginning of the test), the bandwidth consumption in the second run decreases to an average of 500 Kbits per minute against a baseline traffic of 2.2 Mbits per minute measured during the first run. This reduction is caused by two factors: first, the absence of query messages, which do not need to be forwarded between federates when proactive querying is active; and second, a reduction in the amount of data sent, since updates are limited to the resources that have changed state. Note that proactive querying also reduces the latency of requests to values very close to those shown in our second experiment, as all client requests for assets updated proactively will be resolved by MARGOT from the local database.

VI. Related Work

IoT resources discovery within Smart Cities is an active research topic. The authors of [14] performed a comprehensive survey of discovery technologies for IoT environments, analyzing and comparing solutions such as multicast DNS (mDNS), multicast CoAP, the Simple Service Discovery Protocol (SSDP), and others. In [15], the authors present the Smart and Power Efficient Node Discovery Protocol (SPEND), a reliable and energy-efficient discovery mechanism for IoT-Fog networking scenarios that leverages the MQTT protocol to keep track of Things, which act like publishers/advertisers in

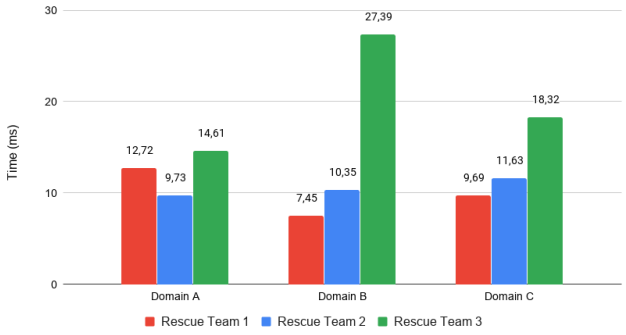

Fig. 4. Average Response Latency per Domain, MARGOT Caching Enabled

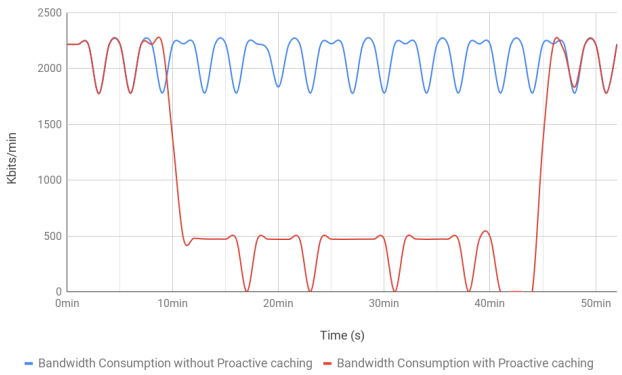

Fig. 5. Bandwidth Consumption: Reactive Querying vs. Proactive Querying

the network. Experimental results show the effectiveness and low power consumption of the protocol, thereby supporting the thesis that MQTT is a good choice for constrained devices. In another study, the authors evaluate a multi-domain, distributed discovery mechanism for the IoT that takes advantage of the CoRE Resource Directory (RD) and CoAP as the interface for discovering and accessing resources [16]. In the this solution, IoT Gateways are responsible for implementing the RD within the single IoT domain, while an approach based on a Distributed Hash Table (DHT) enables global discovery across multiple IoT domains.

A few works also evaluate the importance of resource discovery in the context of Information-centric Networking (ICN), often focusing on the network edge. The work in [17] discusses the possibility of adopting semantic matching techniques to perform IoT service discovery in ICN scenarios to increase the flexibility of the discovery processes. In [18], the authors propose the use of a service-response model to perform resource discovery at the network edge in Named Data Networking (NDN)-based scenarios. The proposed model naturally takes advantage of interest-based routing of NDN, which the authors extend by adding a deferral scheme to avoid collisions over the same service requests. The authors then extend their work in [19], proposing a solution to support resource discovery across NDN-based and IP-based networks. Their approach leverages mDNS to perform discovery inside the IP network and the Named Publish Subscribe Networking (NPSN) protocol within the NDN network; a gateway solution called Future Internet eXchange Point (FIXP) is used to bridge between the different networks and protocols.

MARGOT differs from the aforementioned solutions because it takes advantage of a distributed architecture of gateways that implement domain-specific discovery capabilities and support domain-specific security policies via Federation Services and ABE. Using Federation Services, MARGOT can forward client queries and IoT data over the Federation network based on sharing policies and replicate data to increase its availability, while ensuring that access is given only to authorized users. Furthermore, MARGOT supports pluggable DAs to simplify resource discovery and management in multi-domain scenarios, enabling the use of a wide range of open or closed, standard or non-standard discovery protocols.

VII. Conclusions and Future Work

In this paper, we analyzed the current status of IoT services and traditional solutions adopted to face the problem of IoT discovery in Smart Cities. Then, we discussed future directions of IoT and services, with a special focus on the pivotal role of effective IoT discovery solutions in HADR scenarios. Finally, we described the MARGOT platform and its features, a solution to manage the complexity of IoT asset discovery in multi-domain scenarios and maintain the availability of resource access information in HADR scenarios. Experimental results show that MARGOT can also effectively reduce latency in the IoT discovery process and lower bandwidth consumption under the considered use cases.

In the future, we will add the support for additional IoT services and discovery protocols to MARGOT. We are also planning to enhance the IP to enable more intelligent and effective optimization strategies. These will include a prediction system built upon collected statistics about users' requests and interests to further reduce response times for the users of the system, for instance by anticipating what types of sensors or geographical areas they will be interested in. Furthermore, we are considering further extending the security mechanisms within MARGOT. In particular, we are planning to implement security mechanisms to support different client authentication and authorization policies (owners, admins, guest, et cetera), in order to prevent resource access from ill-intentioned users, and also introduce an authentication system at the level of Federation Services, to avoid that malicious MARGOT instances can successfully federate and inject fake or purposely erroneous discovery data into the system.

References

[1] R. Morello, S. C. Mukhopadhyay, Z. Liu, D. Slomovitz, and S. R. Samantaray, "Advances on Sensing Technologies for Smart Cities and Power Grids: A Review," *IEEE Sensors Journal*, vol. 17, no. 23, pp. 7596–7610, Dec 2017. doi: 10.1109/JSEN.2017.2735539

[2] J. Lin, W. Yu, N. Zhang, X. Yang, H. Zhang, and W. Zhao, "A survey on internet of things: Architecture, enabling technologies, security and privacy, and applications," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1125–1142, Oct 2017. doi: 10.1109/JIOT.2017.2683200

[3] T. Hui, R. Sherratt, and D. Díaz-Sánchez, "Major requirements for building smart homes in smart cities based on internet of things technologies," *Future Generation Computer Systems*, vol. 76, 11 2016. doi: 10.1016/j.future.2016.10.026

[4] M. Noura, M. Atiquzzaman, and M. Gaedke, "Interoperability in internet of things: Taxonomies and open challenges," *Mob. Netw. Appl.*, vol. 24, no. 3, p. 796–809, Jun. 2019. doi: 10.1007/s11036-018-1089-9

[5] P. Barnaghi and A. Sheth, "On searching the internet of things: Requirements and challenges," *IEEE Intelligent Systems*, vol. 31, no. 6, pp. 71–75, Nov 2016. doi: 10.1109/MIS.2016.102

[6] R. Lea and M. Blackstock, "City hub: A cloud-based iot platform for smart cities," in *2014 IEEE 6th International Conference on Cloud Computing Technology and Science*, Dec 2014. doi: 10.1109/CloudCom.2014.65. ISSN null pp. 799–804.

[7] A. Taherkordi and F. Eliassen, "Scalable modeling of cloud-based iot services for smart cities," in *2016 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)*, March 2016. doi: 10.1109/PERCOMW.2016.7457098. ISSN null pp. 1–6.

[8] H. Kim and E. A. Lee, "Authentication and authorization for the internet of things," *IT Professional*, vol. 19, no. 5, pp. 27–33, 2017. doi: 10.1109/MITP.2017.3680960

[9] L. Campioni, R. Lenzi, F. Poltronieri, M. Pradhan, M. Tortonesi, C. Stefanelli, and N. Suri, "MARGOT: Dynamic IoT Resource Discovery for HADR Environments," in *MILCOM 2019 - 2019 IEEE Military Communications Conference (MILCOM)*, Nov. 2019. doi: 10.1109/MILCOM47813.2019.9021092. ISSN 2155-7578 pp. 809–814.

[10] R. Lenzi, G. Benincasa, E. Casini, N. Suri, A. Morelli, S. Watson, and J. Nevitt, "Interconnecting Tactical Service-Oriented Infrastructures with Federation Services," in *MILCOM 2013 - 2013 IEEE Military Communications Conference*, Nov 2013. doi: 10.1109/MILCOM.2013.123. ISSN 2155-7586 pp. 692–697.

[11] D. Evans, "The Internet of Things How the Next Evolution of the Internet Is Changing Everything," CISCO, Tech. Rep., Apr 2011. [Online]. Available: https://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf

[12] A. Pal, H. K. Rath, S. Shailendra, and A. Bhattacharyya, "Chapter 3 IoT Standardization : The Road Ahead," 2018. doi: 10.5772/intechopen.75137

[13] F. Poltronieri, L. Campioni, R. Lenzi, A. Morelli, N. Suri, and M. Tortonesi, "Secure Multi-Domain Information Sharing in Tactical Networks," in *MILCOM 2018 - 2018 IEEE Military Communications Conference (MILCOM)*, Oct 2018. doi: 10.1109/MILCOM.2018.8599693. ISSN 2155-7578 pp. 1–6.

[14] A. Bröring, S. K. Datta, and C. Bonnet, "A Categorization of Discovery Technologies for the Internet of Things," in *Proceedings of the 6th International Conference on the Internet of Things*, ser. IoT'16. New York, NY, USA: Association for Computing Machinery, 2016. doi: 10.1145/2991561.2991570. ISBN 9781450348140 p. 131–139.

[15] Venanzi *et al.*, "MQTT-Driven Sustainable Node Discovery for Internet of Things-Fog Environments," in *2018 IEEE International Conference on Communications (ICC)*, May 2018. doi: 10.1109/ICC.2018.8422200. ISSN 1938-1883 pp. 1–6.

[16] G. Tanganelli, C. Vallati, and E. Mingozzi, "Edge-Centric Distributed Discovery and Access in the Internet of Things," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 425–438, Feb 2018. doi: 10.1109/JIOT.2017.2767381

[17] Quevedo *et al.*, "On the Application of Contextual IoT Service Discovery in Information Centric Networks," *Comput. Commun.*, vol. 89, no. C, p. 117–127, Sep. 2016. doi: 10.1016/j.comcom.2016.03.011

[18] M. Amadeo, C. Campolo, and A. Molinaro, "NDNe: Enhancing Named Data Networking to Support Cloudification at the Edge," *IEEE Communications Letters*, vol. 20, no. 11, pp. 2264–2267, Nov. 2016. doi: 10.1109/LCOMM.2016.2597850

[19] Quevedo *et al.*, "Internet of things discovery in interoperable information centric and IP networks," *Internet Technology Letters*, Jul. 2017. doi: 10.1002/itl2.1